Status	Finished
Started	Tuesday, 19 November 2024, 3:58 PM
Completed	Tuesday, 19 November 2024, 5:07 PM
Duration	1 hour 9 mins

Question 1

Correct

Marked out of 5.00

Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

Get the ASCII values of all the extracted alphabets.

Calculate sum of those ASCII values. Lets call it sum1 and calculate single digit sum of sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Return that single digit as output.

Note:

1. Array size ranges from 1 to 10.
2. All the array elements are lower case alphabets.
3. Atleast one common alphabet will be found in the arrays.

Example 1:

input1: {'a', 'b', 'c'}

input2: {'b', 'c'}

output: 8

Explanation:

'b' and 'c' are present in both the arrays.

ASCII value of 'b' is 98 and 'c' is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

Input	Result
a b c b c	8

Answer: (penalty regime: 0 %)

```
import java.util.HashSet;
import java.util.Scanner;

public class CommonCharacterASCII {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        String input1 = scanner.nextLine();
        char[] charArray1 = input1.replace(" ", "").toCharArray(); // Convert to char array

        // Prompt the user for the second input

        String input2 = scanner.nextLine();
        char[] charArray2 = input2.replace(" ", "").toCharArray(); // Convert to char array

        // Calculate the result
        int result = calculateSingleDigitSum(charArray1, charArray2);
        System.out.println( result); // Display the result

        scanner.close();
    }

    public static int calculateSingleDigitSum(char[] input1, char[] input2) {
        // Step 1: Find common characters
        HashSet<Character> commonChars = new HashSet<>();
        for (char c1 : input1) {
            for (char c2 : input2) {
                if (c1 == c2) {
                    commonChars.add(c1);
                }
            }
        }

        // Step 2: Calculate the sum of ASCII values of common characters
```

```
        // Step 2: Calculate the sum of ASCII values of common characters
        int sum1 = 0;
        for (char c : commonChars) {
            sum1 += (int) c; // Get ASCII value
        }

        // Step 3: Calculate single digit sum
        return getSingleDigit(sum1);
    }

    private static int getSingleDigit(int sum) {
        while (sum >= 10) {
            int tempSum = 0;
            while (sum > 0) {
                tempSum += sum % 10; // Add last digit
                sum /= 10; // Remove last digit
            }
```

	Input	Expected	Got	
✓	a b c b c	8	8	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of 5.00

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case_option parameter, as follows:

If case_option = 0, normal reversal of words i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "orpiW seigoloNhceT eroLagnaB".

If case_option = 1, reversal of words with retaining position's case i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "Orpiw SeigOlonhcet EroIaGnab".

Note that positions 1, 7, 11, 20 and 25 in the original string are uppercase W, T, N, B and L.

Similarly, positions 1, 7, 11, 20 and 25 in the new string are uppercase O, S, O, E and G.

NOTE:

1. Only space character should be treated as the word separator i.e., "Hello World" should be treated as two separate words, "Hello" and "World". However, "Hello,World", "Hello;World", "Hello-World" or "Hello/World" should be considered as a single word.

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is "Wipro TechNologies, Bangalore" the new reversed sentence should be "Orpiw ,seiGolonhceT Erolagnab". Note that comma has been treated as part of the word "Technologies," and when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words "Wipro and Bangalore" have changed to "Orpiw" and "Erolagnab".

3. Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

Examples:

S. No.	input1	input2	output
1	Wipro Technologies Bangalore	0	orpiW seigolonhceT erolagnaB
2	Wipro Technologies, Bangalore	0	orpiW ,seigolonhceT erolagnaB
3	Wipro Technologies Bangalore	1	Orpiw Seigolonhcet Erolagnab
4	Wipro Technologies, Bangalore	1	Orpiw ,seigolonhceT Erolagnab

For example:

Input	Result
`Wipro Technologies Bangalore` `0`	`orpiW seigolonhceT erolagnaB`
`Wipro Technologies, Bangalore` `0`	`orpiW ,seigolonhceT erolagnaB`
`Wipro Technologies Bangalore` `1`	`Orpiw Seigolonhcet Erolagnab`
`Wipro Technologies, Bangalore` `1`	`Orpiw ,seigolonhceT Erolagnab`

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;
import java.util.Arrays;

public class ReverseWords {
    public static String reverseWords(String sentence, int caseOption) {
        String[] words = sentence.split("\\s+"); // Split by whitespace, including commas
        StringBuilder reversedSentence = new StringBuilder();

        for (String word : words) {
            char[] chars = word.toCharArray();
            char[] a=word.toCharArray();
            int start = 0, end = chars.length - 1;

            // Reverse word characters
            while (start < end) {
                char temp = chars[start];
                chars[start] = chars[end];
                chars[end] = temp;
                start++;
                end--;
            }
            int index=chars.length;
            if (caseOption == 1) {
                for (int i = 0; i < chars.length; i++) {
```

```
                    if (Character.isLetter(chars[i])&&i!=index) {
                        chars[i] = Character.isUpperCase(a[i]) ? Character.toUpperCase(chars[i]) : Character.toLowerCase(cha
                    }
                    else  if(!Character.isLetter(chars[i])&&!Character.isWhitespace(chars[i])){
                        char c=a[i];
                        String d=String.valueOf(chars);
                        index=d.indexOf(c);
                        if(index!=-1)
                        chars[index]=a[i];

                      }


                }
            }

            reversedSentence.append(String.valueOf(chars)).append(" ");
        }
        return reversedSentence.toString().trim();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String sentence = scanner.nextLine();
        int caseOption = scanner.nextInt();
        String reversedSentence = reverseWords(sentence, caseOption);
        System.out.println(reversedSentence);
    }
```

	Input	Expected	Got	
✓	Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB	orpiW seigolonhceT erolagnaB	✓
✓	Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB	orpiW ,seigolonhceT erolagnaB	✓
✓	Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab	Orpiw Seigolonhcet Erolagnab	✓
✓	Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab	Orpiw ,seigolonhceT Erolagnab	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

You are provided with a string which has a sequence of 1's and 0's.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000

T : 0000000

and so on upto A having 26 0's (00000000000000000000000000).

The sequence of 0's in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 00001000000000000000000100000000000100000000010000000000001

The decoded string (original word) will be: WIPRO

Note: The decoded string must always be in UPPER case.

For example:

Input	Result
010010001	ZYX
00001000000000000000000100000000000100000000010000000000001	WIPRO

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

public class Decoder {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.nextLine(); // Read the input string of 1's and 0's
        sc.close();

        // Decode the string
        String decodedWord = decode(input);
        System.out.println(decodedWord);
    }

    private static String decode(String input) {
        String[] segments = input.split("1"); // Split by '1'
        StringBuilder decoded = new StringBuilder();

        for (String segment : segments) {
            int length = segment.length();
            if (length > 0 && length <= 26) { // Ensure the length is valid
                // Calculate the corresponding character
                char letter = (char) ('Z' - (length - 1));
                decoded.append(letter);
            }
        }

        return decoded.toString();
    }
}
```

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

◄ Lab-12-MCQ

Jump to...

Identify possible words ►